

Early Evaluation of Security Functionality in Software Projects - some Experience on using the Common Criteria in a Quality Management Process

By Tobias Scherner and Lothar Fritsch

Tobias Scherner is a researcher on multilaterally secure mobile applications and security assurance at the Chair for M-Commerce & Multilateral Security at Johann Wolfgang Goethe-University, Grärfstraße 78, 60054 Frankfurt (tel. +49 69 79825301; fax: +49 69 79825306; e-mail: Tobias.Schermer@M-Lehrstuhl.de).

Lothar Fritsch is a researcher in privacy-friendly location-based services at the Chair for M-Commerce & Multilateral Security at Johann Wolfgang Goethe-University, Grärfstraße 78, 60054 Frankfurt (tel. +49 69 79825301; fax: +49 69 79825306; e-mail: Lothar.Fritsch@M-Lehrstuhl.de).

This is a research report of Johann Wolfgang Goethe University, Frankfurt am Main, June 2007.

Abstract—This paper documents the experiences of assurance evaluation during the early stage of a large software development project. This project researches, contracts and integrates privacy-respecting software to business environments. While assurance evaluation with ISO 15408 Common Criteria (CC) within the certification schemes is done after a system has been completed, our approach executes evaluation during the early phases of the software life cycle. The promise is to increase quality and to reduce testing and fault removal costs for later phases of the development process. First results from the still-ongoing project suggests that the Common Criteria can define a framework for assurance evaluation in ongoing development projects.

I. INTRODUCTION

There exist several approaches to ensure the quality of secure software. Some of these approaches have the focus of quality assurance at a very early stage of the development process and have weaknesses to ensure the quality of this process until the product is ready to enter the market. Other approaches, like the CC, focus on inspection, or more concrete evaluation, of ready-to-market products. We tried to introduce an inspection process that is inspired by the CC evaluation scheme to earlier phases of the software engineering process. Our newly developed approach tries to bridge the gap between requirements engineering, code production and post-evaluation. This is motivated by two effects we expect: First, faults discovered earlier can be removed faster, and second, they can be removed cheaper. To show our point, we first have a look at testing, verification and validation literature from the software engineering field on knowledge. Then we will briefly introduce the Common Criteria scheme. Following this, we describe our process approach to detect security assurance problems in the ongoing development process. In the end, we

give some first experience from the process application in a large security software development project.

A. Cost of Testing

First, we will deal with the question whether early testing efforts in secure software development are economically justified or not. Early testing introduces cost into the design phase - and it might not be trivial to find evidence whether it is worth the investment.

In the literature, one can clearly identify that early fault removal is more economic than late fault removal. Although on first sight, one might conclude that early testing and validation simply shifts testing cost to designers and developers, some economic evidence exists that due to network externalities, code re-use and the software engineering process, early failure detection is notably cheaper than later failure removal. In [1], the cost of fault removal during different phases of software engineering increase exponentially as listed in Table 1.

Phase	Cost
Requirements	10 \$
Analysis	20 \$
Design	30 \$
Code	50 \$
Testing	200 \$
Install	800 \$
End User	1500 \$

Table 1: Cost of fault removal in software engineering according to [1].

Here, early fault removal clearly is much cheaper than later fault removal..

An economic model of bug removal is constructed in [2], where the authors gather evidence for the argument that early bug removal is more efficient than later testing and removal.

B. Testing, Verification and Validation

In this part, we focus on fault prevention rather than fault correction. We looked at several approaches to deal with testing. The United States of America National Aeronautics and Space Administration (NASA) has a strict standard on software quality [3]. In section 3.2.1.2.1 of the document, the mission of software assurance is defined in this way: “A strategy that emphasizes prevention, not correction”. In [4], a consulting firm suggests to use CC elements for early software validation due to the fact that the CC provide a large variety of standardized information and processes on security vulnerabilities. An example of using the CC during a software

development process can be found in [5], where a Palm PDA software has been developed using a process based on the CC requirements.

C. Common Criteria

The Common Criteria for IT Security Evaluation, short CC, provide a collection of generic components of security requirements to aid in the specification of product or system security attributes. The current version 2.21 is similar to the ISO (International Organization for Standardization) standard 15408. The traditional utilization of the CC is the usage as the basis for evaluations of security properties of IT-systems and software. The main objective of the CC, besides a well known and expected standard, is the evaluation of products. This can, among other purposes, be used to provide users and customers a decision support base if this evaluated object meets the own requirements. Examples for evaluated Products are Smartcards from the credit card sector².

The CC advise to produce Protection Profiles (PP) and Security Targets (ST). PP's are an implementation-independent set of security requirements for a category (application specific) of Target of Evaluations (TOE) that meet specific consumer needs. On the other hand ST's are an implementation-dependent set of security requirements and specifications used as the basis for evaluation of the identified TOE. An ST can be compared to the corresponding PP's to assess whether the postulations of the PP are met.

Preferably, the CC shall support the developers to meet the postulated requirements right from the beginning of the development process. But until now this policy is not a formal defined part of the ISO 15408 standard.

II. EARLY SECURITY VALIDATION WITH CC

Our approach is to adapt the principles of the CC of building PP's and ST's during the development process without the standardized components of the CC, but properly reflecting the security requirements which have been defined for the project products. The comparison of ST and PP already during the development revealed different lacks which have been reported to the developers to solve the problems until the next evaluation loop. From the perspective of the project, this early involvement of evaluators offered the chance to fix problems with a lower cost, effort and to fulfill the high self-expectations and the expectations of the commission and the future users.

A. Evaluation and the Common Criteria

The basis of the evaluation process is the current official version 2.2 of the Common Criteria (CC, IS 15408). Essential for developers is the reading of the "Common Methodology for Information Technology Security Evaluation" [6]. This document describes the methodology of different evaluation assurance levels (EAL) including lists of necessary activities.

Following the methodology of the CC the assurance through evaluation has several meanings, and the following list can be seen as a basis of the CC evaluation [7]:

- a) analysis and checking of process(es) and procedure(s);
- b) checking that process(es) and procedure(s) are being applied;
- c) analysis of the correspondence between Target of Evaluation (TOE) design representations;
- d) analysis of the TOE design representation against the requirements;
- e) verification of proofs;
- f) analysis of guidance documents;
- g) analysis of functional tests developed and the results provided (by the software developer);
- h) independent functional testing;
- i) analysis for vulnerabilities (including flaw hypothesis);
- j) penetration testing.

The process of the evaluation is an integrated process over the whole life cycle including the planning of a software project, developing and integrating of components, installing and using the software. So, the above listed elements of an evaluation are far from being complete, but the different evaluation assurance levels extend the evaluation basis by the assurance aspects described in [7].

The evaluation of the project components is not bound to certain evaluation levels and all the formal regulations, but developers and evaluators have to agree on a defined level. From the evaluation point of view the general conditions should follow the requirements of the evaluation level 4. This recommendation is caused by the project technical design principles that state very clearly that the maximum of privacy shall be achieved and to ensure that the principles are fulfilled we need a high level of assurance.

However, the discussion about which level of assurance is needed has to be initiated before the next evaluation cycle starts and we want to invite everybody to contribute to this process. Nevertheless the contribution of a delegation of the evaluators is mandatory to come to an agreement. As an example, the required assurance level for electronic signatures under the electronic signature directive is EAL4+, while a smart card reader for patient data is only tested according to EAL3.

B. Experience with CC based project evaluation

The first cycle of the assurance evaluation according to the Common Criteria (CC) could not be performed yet for the version 1 prototype in its early stage. This was caused by several reasons. First, our analysis showed that the discrepancy between the needed and the available documentation was too high. We investigated this phenomenon and came to the conclusion that developers and evaluators have a different view on what an evaluation is. This is a commonly observable problem while having teams of specialists in different domains cooperating on projects. An interesting approach is to use a prototype as a boundary object to come to a common perspective about the requirements regarding the prototype [8]. To build a boundary object for evaluation could be a great chance for the project to reach to consent about the scope and

¹ Common Criteria Project: *The Common Criteria, Version 2.2, 2004, similar to IS 15408: 2004.*

² A list of PP's and evaluated products can be found under www.commoncriteriaportal.org.

to agree about the boundary conditions of evaluations within the project.

Moreover, the assurance evaluators detected discrepancies between different statements provided by the component developers and the integrators about the implementation stage of security functionalities during the preparation phase of the evaluation. This problem seems to be caused by two associated circumstances. The starting points were integration problems which resulted in deviations from the integration time plan. Thus, the deviation created stress and inhibited adequate communication between component developers and integrators. Thereby, the component developers had no updated information whether their component was integrated or not.

Secondly, the implemented security functionalities of prototype version 1 are not as fully implemented as would have been necessary for a successful assurance evaluation. Especially the lack of some basic security functionality which was omitted for undocumented reasons were strong points of critique.

Of primary importance were the questions how to deal with the inaccurate documentation and the lack of important security functionalities. Facing these problems, the assurance evaluators came to the decision of suspending the evaluation process and instead starting to prepare the evaluation process of version 2, and educating the developers better about assurance preconditions.

We will now describe the pre-conditions that must be fulfilled by the different parties to enable the prototype to enter the evaluation process to avoid future confusions. This guidance is intended for programmers, documentation writers and project managers that work on component design and implementation, or on integration. In the broader sense, this is also the path for the preparation of the evaluation processes of the future prototypes. We expect at least two more cycles of security evaluation before the development is finished.

C. Basic Preconditions for an Evaluation

This section describes the basic needed preconditions for an evaluation of the project software in general, but with the focus on the integrated prototype. Under the notion "precondition" we summarize all documentation that an evaluator needs to accomplish a basic evaluation process in an integrated manner like it is described above.

The following sections describe in detail which documentation an evaluator will expect for:

Implemented security functions.

- Threat analysis, countermeasures, strength of the implementation.
- Test plans.
- Best practice examples for the application prototype on how to use the provided interfaces.

D. Implemented Security Functions

An evaluation requires a list of the implemented security functionalities. This includes on the component level a list of what kind of security functionalities are implemented including the specification (e.g. kind of encryption algorithm, description of the distribution of the keys and the storage),

which countermeasure secures against what kind of threat in which expected strength.

On the level of the prototype, a description of the interaction of the different components is mandatory.

E. Threat Analysis

Threat and vulnerability analyses are one of the most important parts of the preparation material for an evaluation.

The approach of a vulnerability analysis is to find weaknesses of the security of a system or parts of the system.

The threat analysis is based on the perceptions of the vulnerability and characterizes the possible effects of the found weaknesses. The documentation empowers the evaluators to understand the background of implementations and to come to an assessment if the known possible threats can be counter measured by the implemented security functions. Following the CC part 3 [7] vulnerabilities can arise through failures in:

- Requirements – that is, an IT product or system may possess all the functions and features required of it and still contain vulnerabilities that render it unsuitable or ineffective with respect to security;
- Construction – that is, an IT product or system does not meet its specifications and/or vulnerabilities have been introduced as a result of poor constructional standards or incorrect design choices;
- Operation – that is, an IT product or system has been constructed correctly to a correct specification but vulnerabilities have been introduced as a result of inadequate control upon the operation of it.

A possible, and from our point of view, adequate presentation of a threat analysis can be found below.

Example: communication

List of components

Component's name:	Component's number:	Interacts with the following components:	Description:
communication	C_1	Event manager	Responsible for the communication between the users, service providers and internal communication.

List of threats

Number of the threat:	Description:
T_1	Communication can be eavesdropped (and analysis provides meaningful results).
T_2	Communication partners can be revealed to a third party
T_3	Communication can be altered
T_4	Communication partners can forge their identity.
T_5

List of countermeasures

Number of counter-measure:	Description of countermeasure:	Eases impact of threat number:	Strength: (low / medium / high)
CM_1	Use of encryption mechanism like 3DES and AES	T_1, T_3	High
CM_2	Use of authentication mechanism like certificates	T_4	Medium
CM_3	Use of Mixes and dummy traffic	T_2	Medium / High

F. Test Plans

Test plans have multiple dimensions. The first dimension concerns the components, the integration and the system as it is for example described in [9]. Each of these levels has to be tested and the tests have to be documented.

The second dimension covers the testing of security functionalities, tests of the interfaces to later on used parts of the project and handling of unexpected situations (e.g. test of stability of the programs if these programs are contacted with unexpected enquiries).

The documentation of the tests covers:

- The character of the conducted test (e.g. functionality, security or stability test).
- Scope of the test (e.g. tested components, interaction with other parts of the project software).
- The documentation of the test procedure. This includes the test configuration including the used tools and the underlying infrastructure inclusive test criteria and conditions that describe why tests have been terminated.

G. The documentation of the test results.

A suitable test standard is the IEEE standard “829-1998 IEEE Standard for Software Test Documentation” [10] which accurately describes the composition of test plans and offers standardized documents to support the efficiency of the test team and additionally the evaluators.

H. Enforcement of the Evaluation

We will evaluate the next versions of the prototype by using the following evaluation model. In this section we describe why this approach was changed for the integrated prototype version 1.

1) Process One

The starting point is the test release of the prototype Version 2. This provides an overview of the security and privacy functionalities. The next step is to identify the integrated components. For each component we will do an examination of its contribution to privacy and security.

This contains in detail:

What is the purpose of the component (e. g. what the benefit of the implementation for the end-user is)? The main sources for this are the project- architecture-deliverables.

What are the possible threats? We will do such an analysis for the input of the developers and create our own threat approach. For each privacy goal, there might exist several threats. Hence, we want to summarize how the targeted benefit of each component can be weakened or totally neutralized through different threats. This detailed analysis considers the fact that a system is only as strong as its weakest part.

For the last two items we need input from the developers of the components, who provide their threat analysis and countermeasures as described above. The approach do create our own threat analysis may lead to a more complete presentation.

The next step is to analyze the specifications. The purpose is to evaluate if the provided functionalities can deal with the investigated threats. This should result in a first indication of whether the prototype fulfils the claimed requirements or not.

To be able to compare the investigated requirements we have to build a security target (ST) for the integrated prototype.

2) Process Two

Starting from the requirements postulated in the requirements deliverable, the evaluators have to summarize and structure the requirements regarding the integrated prototype. In the first iteration this will not be as formalized as it is claimed in the Common Criteria. This will be a further step towards creating protection profiles.

Further on, the next task is to create a lightweight Protection Profile (PP). The notation “lightweight” was chosen, because it may not fit the formalized requirements of the Common Criteria provided that the postulated requirements would apply one-to-one without transformation into the structure of functional components of the CC. So, the lightweight PP will reflect the basic requirements [11] like unlinkability, pseudonymity, repudiation building and anonymous communication in natural language and it will provide a TSF (TOE Security Functionality) description according to the CC.

3) Joint Process

To combine the two previous parallel processes the evaluators have to compare the Protection Profile of the users’ point of view and the security target of the components. At this point the evaluators have to analyze if the postulations of the protection profile meet the requirements of the security target. This operation can be understood as a mapping of the two constructs. Due to the deviation of the lightweight PP from the formalized requirements of the CC the mapping is more a global examination whether the ST claims conformance with the PP than a real conformance check. Nevertheless, this should lead to an assessment in how far the prototype meets the postulated requirements. At the end of this joint process it is possible to get to a conclusion about the quality of implementation of the integrated prototype.

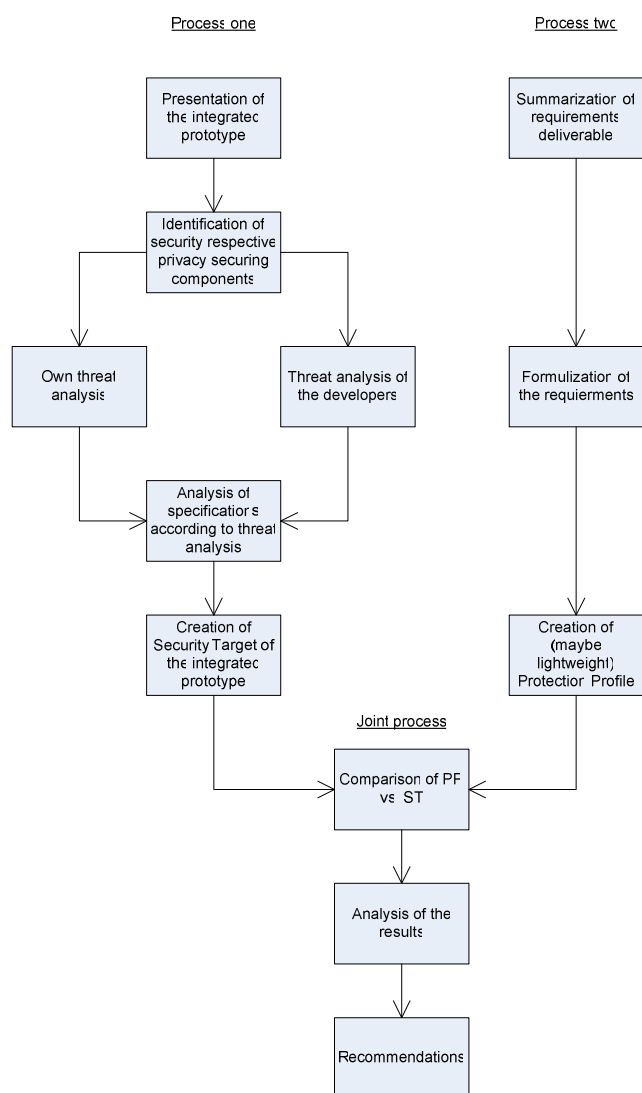


Figure 1 Process to evaluate the integrated prototype

III. CONCLUSION

Even without successfully conducting the first evaluation cycle, the main conclusion of the iteration of the assurance evaluation process is that the developers had difficulties to meet the expectations of the evaluators. Overall threat models, security mechanisms and code re-use analysis was not done. Some components had nothing but a claim about their security functionality, but no documentation. They missed to document their threat and risk analysis and had to face many integration difficulties. In addition the lack of communication among the developers and between developers of the components and integrators, this led to a dissatisfying first judgment about the current step of implementation. The suggestions of the evaluators are that the developers have to follow a more formal process regarding analysis, specification, developing and documentation. They should do more reflection on their work to discover inconsistencies during their decisions and not at the stage of delivering the prototypes to the evaluators.

Without our evaluation approach, we would not have found many problems at this early stage. This also allows the developers to meet the project time plans and quality demands until the end of the project. A traditional CC evaluation would have brought up these problems at the end of the project, which would have endangered the success of the whole project beyond its deadline.

Our first application of the CC based early evaluation process discovered many design and documentation inconsistencies and surfaced several implementation problems. It therefore can be regarded as a success. After our next step – education of developers about accurate analysis and documentation – we are looking forward the next assurance cycle in December 2005 to get deeper insight in the usefulness of our evaluation process. The results so far suggest that it supports early security fault detection and removal, which according to section I.A will lead to lower cost of the software engineering process.

Much work has yet to be done. After a few more applications of our process to software development, the economic effects of its application should be monitored in a real project. Also, some research among the developers about the cost of educating and motivating them to model according to CC requirements should be performed. Finally, modeling our CC based approach into a procedure like the clean room software development process might lead to a widely applicable model for security assurance by early assurance in software projects. This finally could be compared against other methods of early validation.

IV. LITERATURE

- [1] T. Esko, Quality Assurance in Corporate IT: It Matters More than Ever. Los Angeles, USA: 2001.
- [2] R. L. Vienneau, "The Present Value of Software Maintenance," Journal of Parametrics. vol. 15, pp. 18-36, 1 1995.
- [3] F. Gregory, SOFTWARE ASSURANCE STANDARD NASA-STD-2201-93. Washington D.C.: 1992.
- [4] R. Exler, Security and the Application Development Process. 2004.
- [5] M. Vetterling, G. Wimmel and A. Wisspeintner, "Secure systems development based on the common criteria: the PalME project," SIGSOFT Softw. Eng. Notes. vol. 27, pp. 129-138, 6 2002.
- [6] Common Criteria Project, Common Methodology for Information Technology Security Evaluation, Version 2.2, similar to IS 18405. 2004.
- [7] Common Criteria Project, The Common Criteria Part 3. 2004.
- [8] J. Gunaratne, B. Hwong, C. Nelson and A. Rudorfer, Using Evolutionary Prototypes to Formalize Product Requirements. Edinburgh, Scotland : 2004.
- [9] R. C. Rocha and E. Martins, "A strategy to improve component testability without source code," in Testing of Component-Based Systems and Software Quality, S. Beydeda, V. Gruhn, J. Mayer, R. Reussner and F. Schweiggert, Ed. Bonn: Köllen Druck+Verlag GmbH, 2004, pp. 47 - 62.
- [10] IEEE Standards Association, 829-1998 IEEE Standard for Software Test Documentation. IEEE, 1998.
- [11] PRIME Project, Requirements Version 0 – Part 3: Application Requirements. 2004.

Acknowledgements

Part of this research was funded by the European Union within its IST PRIME project. Please refer to www.prime-project.eu for further reference. However note that this paper reflects the authors' opinions only.